Scaffolding Answer Explanation in a Data Normalization Tutor

Antonija MITROVIC

Intelligent Computer Tutoring Group

Computer Science Department, University of Canterbury

Private Bag 4800, Christchurch, New Zealand

tanja@cosc.canterbury.ac.nz

Phone (64) 3 3642987 ext 7771

Abstract: Self-explanation is one of the most effective learning strategies, resulting in deep knowledge. In this paper, we discuss how NORMIT supports self-explanation. NORMIT is a constraint-based tutor that teaches data normalization, a procedural task. We present the system first, and then discuss how it supports self-explanation. We hypothesized the self-explanation support in NORMIT will result in increased problem solving skills and better conceptual knowledge. An evaluation study of the system was performed in October 2003, the results of which show that both problem-solving performance and the understanding of the domain of students who self-explained increased. We also discuss our plans for future research.

1. Introduction

The goal of intelligent educational systems is to support students' learning, and yet evaluations show that even in the most effective systems, some students acquire shallow knowledge. Examples include situations when the student can guess the correct answer, instead of using the domain theory to derive the solution. Aleven et al. [1] illustrate situations when students guess the sizes of angles based on their appearance. On the other hand, we want students to acquire deep, robust knowledge, which they can use to solve different kinds of problems, and to develop effective meta-cognitive skills.

One of the approaches to acquiring deep knowledge is to self-explain. Psychological studies [6,7] show that self-explanation is one of the most effective learning strategies. In self-explanation, the student solves a problem (or explains a solved problem) by specifying why a particular action is needed, how it contributes toward the solution of the problem, and what basic principles of the domain were used to perform the action. Self-explanation has been supported in several existing intelligent tutoring systems with extremely good results [1-4,8].

This paper presents the support for self-explanation in NORMIT, a data normalization tutor. Section 2 reviews related work. Section 3 overviews the learning task, while the architecture of the system is given in Section 4. Support for self-explanation is discussed in Section 5. The results of an evaluation study of NORMIT are presented in Section 6. Finally, the conclusions and avenues for future research are given in the final section.

2. Related Work

Metacognition includes processes involved with awareness of, reasoning and reflecting about, and controlling one's cognitive skills and processes. Metacognitive skills can be taught [5], and result in improved problem solving and better learning [1,8]. Of all metacognitive skills, self-explanation has attracted most interest within the ITS community.

By explaining to themselves, students integrate new knowledge with existing knowledge. Furthermore, psychological studies show that self-explanation helps students to correct their misconceptions [7]. Although many students do not spontaneously self-explain, most will do so when prompted [6] and can learn to do it effectively [5].

SE-Coach [8] is a physics tutor that supports students while they study solved examples. The authors claim that self-explanation is better supported this way, than asking for explanation while solving problems, as the latter may put too big a burden on the student. In this system, students are prompted to explain a given solution for a problem. Different parts of the solution are covered with boxes, which disappear when the mouse is positioned over them. This masking mechanism allows the system to track how much time the student spends on each part of the solution. The system controls the process by modelling the self-explanation skills using a Bayesian network. If there is evidence that the student has not self-explained a particular part of the example, the system will require the student to specify why a certain step is correct and why it is useful for solving the current problem. Empirical studies performed show that this structured support is beneficial in early learning stages.

On the other hand, Aleven and Koedinger [1] explore how students explain their own solutions. In the PACT Geometry tutor, as students solve problems, they specify the reason for each action taken, by selecting a relevant theorem or a definition from a glossary. The performed evaluation study shows that such explanations improve students problem-solving and self-explanation skills and also result in transferable knowledge. In Geometry Explanation Tutor [2], students explain in natural language, and the system evaluates their explanations and provides feedback. The system contains a hierarchy of 149 explanation categories [3], which is a library of common explanations, including incorrect/incomplete ones. The system matches the student's explanation to those in the library, and generates feedback, which helps the student to improve his/her explanation.

In a recent project [14], we looked at the effect of self-explanation in KERMIT, a database design tutor [13]. In contrast to the previous two systems, KERMIT teaches an

open-ended task. In geometry and physics, domain knowledge is clearly defined, and it is possible to offer a glossary of terms and definitions to the student. Conceptual database design is a very different domain. As in other design tasks, there is no algorithm to use to derive the final solution. In KERMIT, we ask the student to self-explain only in the case their solution is erroneous. The system decides on which errors to initiate a self-explanation dialogue, and asks a series of question until the student gives the correct answer. The student may interrupt the dialogue at any time, and correct the solution. We have performed an experiment recently, the results of which show that students who self-explain acquire more conceptual knowledge than their peers.

3. Learning Data Normalization in NORMIT

Database normalization is the process of refining a relational database schema in order to ensure that all tables are of high quality [9]. Normalization is usually taught in introductory database courses in a series of lectures that define all the necessary concepts, and later practised on paper by looking at specific databases and applying the definitions.

Like other constraint-based tutors [11,13,14], NORMIT is a problem-solving environment, which complements traditional classroom instruction. The emphasis is therefore on problem solving, not on providing information. After logging in, the student selects a problem to work on.

Database normalization is a procedural task: the student goes through a number of steps to analyze the quality of a database. We described the tasks NORMIT supports in detail elsewhere [10]. NORMIT requires the student to determine candidate keys (Figure 1), the closure of a set of attributes and prime attributes, simplify functional dependencies,

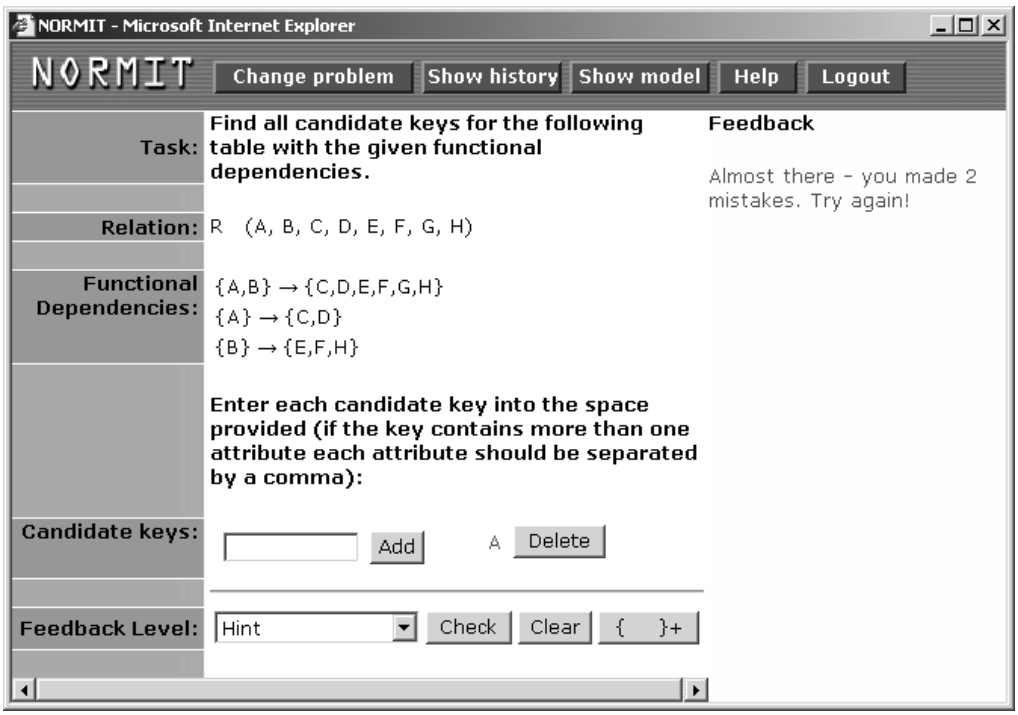

Fig. 1. A screenshot from NORMIT

determine normal forms, and, if necessary, decompose the table. The sequence is fixed: the student will only see a Web page corresponding to the current task. The student may submit a solution or request a new problem at any time. He/she may also review the history of the session, or examine the student model.

When the student submits the solution, the system analyses it and offers feedback. The first submission receives only a general feedback, specifying whether the solution is correct or not. If there are errors in the solution, the incorrect parts of the solution are shown in red. In Figure 1, for example, the student has specified A as the key of the given relation, which is incorrect. On the second submission, NORMIT provides a general description of the

error, specifying what general domain principles have been violated. On the third submission, the system provides a more detailed message, by providing a hint as to how the student should change the solution. The student can also get a hint for every error (as in Figure 1). The correct solution is only available on request.

4.The Architecture of NORMIT

NORMIT is a Web-enabled tutor with a centralized architecture (Figure 2). All tutoring functions are performed on the server side, where student models are also kept. NORMIT is developed in AllegroServe Web server, an extensible server provided with Allegro Common Lisp. At the beginning of interaction, a student is required to enter his/her name, which is necessary in order to establish a session. The session manager requires the student modeller to retrieve the model for the student, if there is one, or to create a new model for a new student. NORMIT identifies students by their login name, which is embedded in a hidden tag of HTML forms. Each action a student performs is sent to the session manager, as it has to link it to the appropriate session and store it in the student's log. Then, the action is sent to the pedagogical module (PM). If the submitted action is a solution to the current step, PM sends it to the student modeller, which diagnoses the solution, updates the student model, and sends the result of the diagnosis back to PM, which generates feedback.

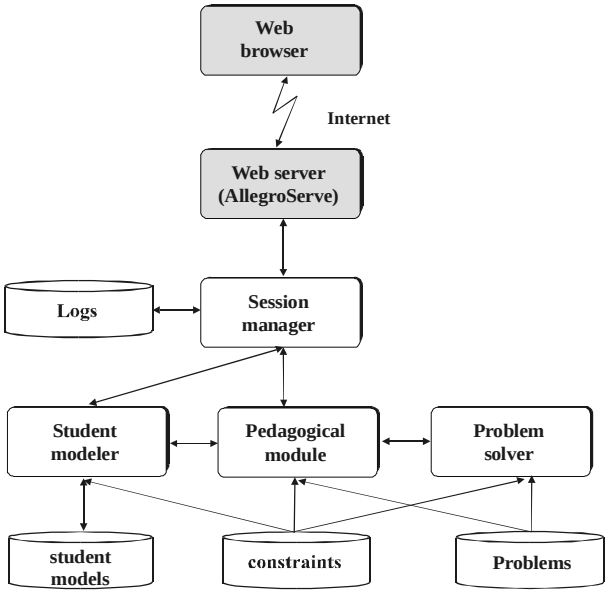

Fig. 2. The architecture of NORMIT

Domain knowledge consists of a set of constraints. Constraint-Based Modeling (CBM) [11,12] is a student modeling approach that is not interested in the exact sequence of states in the problem space the student has traversed, but in what state he/she is in currently. As long as the student never reaches a state that is known to be wrong, they are free to perform whatever actions they please. The domain model is a collection of state descriptions of the form:

If <relevance condition> is true, then <satisfaction condition>

had better also be true, otherwise something has gone wrong.

The constraints are written in Lisp, and can contain built-in functions as well as domain-specific functions. Figure 3 illustrates constraint 10. Each constraint has a unique number, followed by the relevance and satisfaction conditions. The relevance condition of constraint 10 tests whether the current task is the candidate keys task, and then it checks whether the student has specified any candidate keys. Finally, it binds variable *k* to each specified candidate key, thus forming a multiple binding list. The satisfaction part consists of a single test, which is applied to each binding of variable *k*. If the specified candidate key is a superkey, constraint 10 is satisfied. In the opposite case, the student will be given feedback. The next element specified the task the constraint applies to, followed by two feedback messages in each constraint, which are given to the student if his/her solution is incorrect. The first message is shorter, and tells the student what is wrong with the solution.

```
(10   (and (equalp (current-task sol) 'candkeys)(not (null (candkeys sol)))
           (bind-all ?k (candkeys sol) bindings))
      (superkeyp TS (quote ?k) (problem sol))
      "CANDIDATE KEYS"
      “You have specified candidate key(s) incorrectly!"
      "A candidate key you specified does not determine all other
      attributes. The closure of the candidate key must contain all other
      attributes of the relation."
      (?k "candkeys"))
```

Fig. 3. An example constraint

If the student still cannot correct the solution after this message, NORMIT will present the second message, which explains why the specified set of attributes is not a candidate key (as in Figure 1). The last element of the constraint specifies the part of the solution that is incorrect (in this case, that is the attribute to which variable *k* is bound). This binding is used for highlighting the error.

NORMIT currently contains 81 problem-independent constraints that describe the basic principles of the domain. Some constraints check the syntax of the solution, while others check the semantics, by comparing the student's solution to the ideal solution, generated by the problem solver. In order to identify constraints, we studied material in textbooks [9], and also used our own experience in teaching database normalization.

The short-term student model consists of a list of violated and a list of satisfied constraints for the current attempt. The long-term model records the history of usage for each constraint. This information is used to select problems of appropriate complexity for the student, and generate feedback.

5. Supporting Self-Explanation

NORMIT is a problem-solving environment, and therefore we ask students to self-explain while they solve problems. In contrast to other ITSs that support self-explanation, we do not expect students to self-explain every problem-solving step. Instead, NORMIT will require an explanation for each action that is performed for the first time. For the subsequent actions of the same type, explanation is required only if the action is performed incorrectly. We believe that this strategy will reduce the burden on the more able students (by not asking them to provide the same explanation every time an action is performed correctly), and also that the system would provide enough situations for students to develop and improve their self-explanation skills.

Similar to the PACT Geometry Tutor and SE-Coach, NORMIT supports self-explanation by prompting the student to explain by selecting one of the offered options. In

Figure 1, the student specified A as the candidate key incorrectly. NORMIT then asks the following question[1]:

This set of attributes is a candidate key because:

- *It is a minimal set of attributes*
- *Every value is unique*
- *It is a minimal set of attributes that determine all attributes in the table*
- *It determines the values of all other attributes*
- *All attributes are keys*
- *Its closure contains all attributes of the table*

Note that the options offered are not strict definitions from the textbook, and the student needs to reason about them to select the correct one for the particular state of the problem. If the student's explanation is incorrect, he/she will be given another question, asking to define the underlying domain concept (i.e. candidate keys). For the same situation, the student will get the following question after giving an incorrect reason for specifying A as the candidate key:

A candidate key is:

- *an attribute with unique values*
- *an attribute or a set of attributes that determines the values of all other attributes*
- *a minimal set of attributes that determine all other attributes in the table*
- *a set of attributes the closure of which contains all attributes of the table*
- *a minimal superkey*
- *a superkey*
- *a key other than the primary key*
- *A candidate key is an attribute or a set of attributes that determine all other attributes in the table and is minimal. The second condition means that it is not*

[1] The order in which the options are given is random, to minimize guessing

possible to remove any attributes from the set, and still have the remaining attributes to determine the other attributes in the table.

In contrast to the first question, which was problem-specific, the second question is general. If the student selects the correct option, he/she will resume with problem solving. In the opposite case, NORMIT will provide the correct definition of the concept.

In addition to the model of the student's knowledge, NORMIT also models the student's self-explanation skills. For each constraint, the student model contains information about the student's explanations related to that constraint. The student model also stores the history of student's explanation of each domain concept.

6. Experiment

We performed an evaluation study with the students enrolled in an introductory database course at the University of Canterbury. Our hypothesis was that self-explanation would have positive effects on both procedural knowledge (i.e. problem solving skills) and conceptual knowledge. Prior to the experiment, the students had four lectures and one tutorial on data normalization. The system was demonstrated in a lecture on October 14, 2003 (during the last week of the course), and was open to the students a day later. The students in the control group used the basic version of the system, while the experimental group used NORMIT-SE, the version of the system that supports self-explanation. The participation was voluntary, and 31 out of 125 students enrolled in the course used the system. The students were free to use NORMIT when and for how long they wanted.

The pre-test was administered on-line at the beginning of the first session. The post-test was also administered on-line, the first time a student logged on to the system on or after October 29, 2003. The date for the post-test was chosen to be just one day before the exam. We developed two tests, each having four multichoice questions. The first two questions required students to identify the correct solution for a given problem, while for the other

two students needed to identify the correct definition of a given concept. These two tests were randomly used as the pre/post test.

We collected data about each session, including the type and timing of each action performed by the student, and the feedback obtained from NORMIT. Two students who have not attempted any problems, and we excluded their logs from analyses. The results on the pre- and post-tests are given in Table 1.

Table 1. Mean system interaction details

	NORMIT	NORMIT-SE
Students	14	15
Sessions	2.29 (1.38)	1.27 (0.59)
Time spent on problem solving (min.)	144.64 (131.24)	90.27 (166.83)
Attempted problems	14.29 (10.84)	7 (10.13)
Completed problems (%)	54.60 (40.13)	38.29 (46.19))
Pre-test (%)	60.71 (23.44)	55 (31.62)
Post-test (%)	58.33 (8.19)	66.67 (4.43)

The groups are comparable, as there is no significant difference on the pre-test. The average mark on the pre-test for all students was 57.76% (sd = 27.63), while the average for the post-test was 62.5% (sd = 26.22). Therefore, students did improve on the post-test, but the improvement is not significant. Only three students from each group sat the post-test, and we have not analysed the improvements between the groups, as the sample was too small. We believe the small number of submitted post-tests was due to the timing of the study. As stated earlier, NORMIT was demonstrated to the student in the last week of lectures, and they were supposed to use the system on their own, after the lectures were finished. 70% of the students had their first session with NORMIT on or after October 25, 2003, just four days before the post-test. Four students tried the system for the first time on the last day of the study. The longest session was 470 minutes long, and the number of sessions ranged from 1 to 5. The total time spent with NORMIT ranged from only three to 653 minutes. Therefore, the total time they spent with the system is quite low, and significant learning gains cannot be expected.

The number of attempted problems ranged from 1 to 30, with the average of 10.5, and the percentage of solved problems from 1 to 100%. Eleven students attempted some problems, but completed none of them. The remaining students solved at least one problem, with the maximum of 29 solved problems. On average, the control group students had more sessions, spent more time and attempted and solved more problems. However, the experimental group students had improved on the post-test, while the control group did worse than in the pre-test, which seems to support our hypothesis that self-explanation results in improved knowledge.

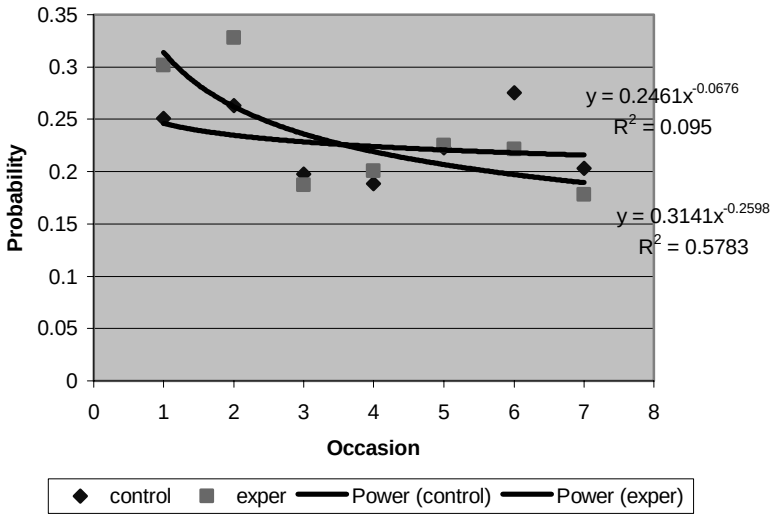

Fig. 4. Learning constraints

Figure 4 shows how students learnt constraints. We looked at the proportion of violated constraints following the *n*th occasion when a constraint was relevant, averaged across all students and all constraints. The R^2 fit to the power curve is good for the experimental group, but is quite poor for the control group students. However, we must note that the number of students was fairly low, thus allowing for statistical effects.

Figure 5.a shows the probability of giving an incorrect answer to the first self-explanation question. As can be seen, this probability varies over occasions, but always

stays quite high, above 0.5. Therefore, students find it hard to give reasons for their actions in the context of the current problem. Some concepts are much more difficult for students to learn than others. For example, out of the total of 49 situations when students who were asked to explain why a set of attributes is a candidate key, the correct answer was given in only 13 cases. Figure 5.b shows the same probability for the question asking to define a domain concept, and the students were better at that. In the case of candidate keys, although students were pretty bad in justifying their choice of candidate key in a particular situation (when the correct answer was given in 26.5% of the cases), when asked to define a candidate key, they were correct in 70% of the cases. The sudden increase at occasion five in Figure 5.b is due to the increasing complexities of the problems students encounter around that time. The likelihood of erroneous answers however continues to decrease after this. Therefore there is evidence that students do improve their declarative knowledge by answering these questions.

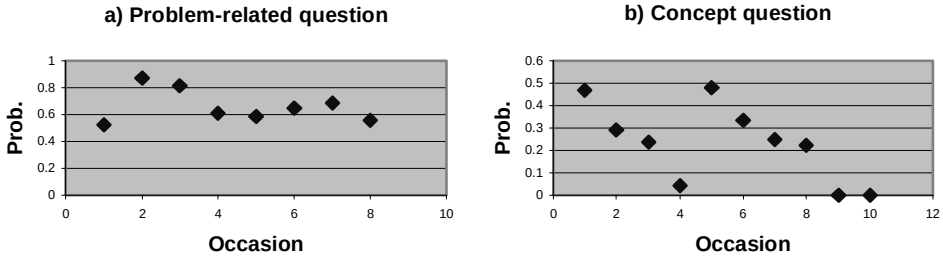

Fig. 5. Student's performance on self-explanation

We also have data on students' performance on a written test, which included one question on data normalization. When comparing the marks of all students who used NORMIT to the other students in the class, there was no significant difference. However, there was a significant difference ($p<0.02$) between the students who have not used NORMIT and the students who have used NORMIT for more than 90 minutes. Due to the small number of participants and the timing of the study, we are not able to compare the

performance of the two groups of students on the post-test, and therefore plan to conduct a bigger study in 2004.

7. Conclusions

Self-explanation is known to be an effective learning strategy. Since intelligent tutoring systems aim to support good learning practices, it is not surprising that researches have started providing support for self-explanation. In this paper, we present NORMIT, a data normalization tutor, and describe how it supports self-explanation. NORMIT is a problem-solving environment, and students are asked to explain their actions while solving problems. The student must explain every action that is performed for the first time. However, we do not require the student to explain every action, as that would put too much of a burden on the student and reduce motivation. NORMIT requires explanations in cases of erroneous solutions. The student is asked to specify the reason for the action, and, if the reason is incorrect, to define the domain concept that is related to the current task. If the student is not able to identify the correct definition from a menu, the system provides the definition of the concept. NORMIT was used in a real course for the first time in 2002, and in 2003 we performed an evaluation study. The results of the study seem to support our hypothesis: students who self-explained improved.

At the moment, the student model in NORMIT contains a lot of information about the student's self-explanation skills that is not used. We plan to use this information to identify domain concepts for which the student needs more instruction. Furthermore, the self-explanation support itself may be made adaptive, so that different support would be offered to students who are poor self-explainer in contrast to students who are good at it. Finally, we plan to perform a bigger evaluation study, in order to be able to assess the effects of the self-explanation support properly.

Acknowledgements: We thank Li Chen and Melinda Marshall for implementing NORMIT's interface.

References

1. Aleven, V., Koedinger, K. R., Cross, K. (1999) Tutoring Answer Explanation Fosters Learning with Understanding. In: Lajoie, S.P. and Vivet, M.(eds), Proc. AIED 1999, IOS Press, 199-206.
2. Aleven, V., Popescu, O., Koedinger, K. R. (2001) Towards Tutorial Dialogue to Support Self-Explanation: Adding Natural Language Understanding to a Cognitive Tutor. *IJAIED*, 12, 246-255.
3. Aleven, V., Popescu, O., Koedinger, K. (2002) Pilot-Testing a Tutorial Dialogue System that Supports Self-Explanation. In: S. Cerri, G. Gouarderes and F. Paraguacu (eds.) Proc. ITS 2002, Springer, LNCS 2363, 344-354.
4. Aleven, V., Koedinger, K., Popescu, O. (2003) A Tutorial Dialogue System to Support Self-Explanation: Evaluation and Open Questions. In: U. Hoppe, F. Verdejo and J. Kay (eds.) Proc. AIED 2003, IOS Press, 39-46.
5. Bielaczyc, K., Pirolli, P., Brown, A.L. (1993) Training in Self-Explanation and Self-Regulation Strategies: Investigating the Effects of Knowledge Acquisition Activities on Problem-solving. *Cognition and Instruction*, 13(2), 221-252.
6. Chi, M. T. H. (2000) Self-explaining Expository Texts: The dual processes of generating inferences and repairing mental models. *Advances in Instructional Psychology*, 161-238.
7. Chi, M.T. (1994) Eliciting self-explanations improves understanding. *Cognitive Science*, 18.
8. Conati, C., VanLehn, K. (2000) Toward Computer-Based Support of Meta-Cognitive Skills: a Computational Framework to Coach Self-Explanation. *Int. J. AI in Education*, 11 389-415.
9. Elmasri, R., Navathe, S.B. (2003) *Fundamentals of database systems*. Benjamin/Cummings, Redwood.
10. Mitrovic, A. (2002) NORMIT, a Web-enabled tutor for database normalization. Kinshuk, R. Lewis, K. Akahori, R. Kemp, T. Okamoto, L. Henderson, C-H Lee (eds) Proc. ICCE 2002, 1276-1280.
11. Mitrovic, A., Ohlsson, S. (1999) Evaluation of a constraint-based tutor for a database language, *Int. J. Artificial Intelligence in Education*, 10(3-4), 238-256.
12. Ohlsson, S. (1994) Constraint-based Student Modeling. In *Student Modeling: the Key to Individualized Knowledge--based Instruction.* Berlin: Springer-Verlag, 167-189.

13. Suraweera, P. and Mitrovic, A., KERMIT: a Constraint-based Tutor for Database Modeling. In: S. Cerri, G. Gouarderes and F. Paraguacu (eds.) *Proc. ITS 2002*, Biarritz, France, LCNS 2363, 2002: 377-387.

14. Weerasinghe, A., Mitrovic, A. (2002) Enhancing learning through self-explanation. Kinshuk, R. Lewis, K. Akahori, R. Kemp, T. Okamoto, L. Henderson, C-H Lee (eds.) Proc. ICCE 2002, 244-248.